Sensors and Algorithms for Small Robot Leader/Follower Behavior

Robert W. Hogg, Arturo L. Rankin, Michael C. McHenry, Daniel M. Helmick,
Charles F. Bergh, Stergios I. Roumeliotis, Larry Matthies

Jet Propulsion Laboratory - California Institute of Technology
4800 Oak Grove Drive, Mail stop 125-209
Pasadena, California 91109

ABSTRACT

Tracked mobile robots in the 20 kg size class are under development for applications in urban reconnaissance. For efficient deployment, it is desirable for teams of robots to be able to automatically execute leader/follower behaviors, with one or more followers tracking the path taken by a leader. The key challenges to enabling such a capability are (1) to develop sensor packages for such small robots that can accurately determine the path of the leader and (2) to develop path-following algorithms for the subsequent robots. To date, we have integrated gyros, accelerometers, compass/inclinometers, odometry, and differential GPS into an effective sensing package for a small urban robot. This paper describes the sensor package, sensor processing algorithm, and path tracking algorithm we have developed for the leader/follower problem in small robots and shows the results of performance characterization of the system. We also document pragmatic lessons learned about design, construction, and electromagnetic interference issues particular to the performance of state sensors on small robots.

Keywords: Mobile robots, path following, leader/follower, pure pursuit, localization, Kalman filtering.

1. INTRODUCTION

Mobile robots that are small and light enough to be carried in a backpack (i.e. "packbots") by individual soldiers have great potential to enhance the safety and effectiveness of urban reconnaissance operations. The size and weight of one recent prototype of such a robot is 60 × 50 × 17 cm and 20 kg.[1] Teams of these robots can be far more effective than individual robots for two reasons. First, a single robot cannot carry all of the sensor and effector payloads required for many missions. Second, multiple robots will often be necessary to cover multiple points of observation.

The need for multiple robots raises the problem of how to navigate them from the departure point to the objective with minimal burden on the human operator. Operator involvement is necessary to designate waypoints and intermediate objectives for the first robot; however, it is desirable for the rest of the robot team to automatically follow the path of the leader, without necessarily maintaining visual contact with each other.

Prior work on robot leader/follower behavior has used a variety of approaches, including visual motion tracking of the lead vehicle[2] and using INS/GPS systems to record the path of the leader, which is then traversed by the follower using the same sensors.[3] Methods depending on visual contact do not meet the needs of our application. Prior path following work based on INS/GPS has all been done on much larger vehicles. Hence, one of the main challenges for packbots has been to identify a sensor suite that would enable path following within the size, weight, power, and cost envelope of our vehicles. Other challenges include coping with GPS dropouts in urban areas and under tree canopies, coping with obstacles that fall within the positional uncertainty of the path following system, and enabling path following through constrictions that require greater positional accuracy than is available from the INS/GPS sensors. An example of the latter is following a path that leads through a culvert.

We are developing a leader/follower system that addresses all of the above challenges. We have completed a path following system based on Kalman filtered IMU, differential GPS, compass/inclinometers, and wheel encoder data. Obstacle avoidance is achieved with an arbiter-based architecture that combines steering votes from the path-following behavior with steering votes from a stereo vision-based obstacle avoidance behavior. We are currently extending this system by (1) using optimal smoothing of the leader's path to reduce the impact of GPS dropouts, (2) developing specialized feature recognition and tracking algorithms to guide the followers through constrictions such as doors and culverts, and (3) developing more general outdoor mapping and landmark recognition capabilities to further reduce the reliance on GPS.

Section 2 describes the navigation sensors we considered, those we integrated, and performance characterization tests done to date on the chosen sensors. Section 3 describes the structure of our pose estimation Kalman filter. Section 4 outlines the architecture of our leader/follower system and describes our path-following control algorithm. Experimental results are shown in section 5. We discuss the significance of the results, highlight open issues, and outline the extensions we have in progress in section 6.

Send correspondence to L. Matthies, Email: larry.matthies@jpl.nasa.gov; Telephone: 818-354-3722; Fax: 818-393-4085; This work was supported by the Tactical Mobile Robotics Program of the Defense Advanced Research Projects Agency (DARPA) Advanced Technology Office under contract NAS 7-1407, task order 15089.

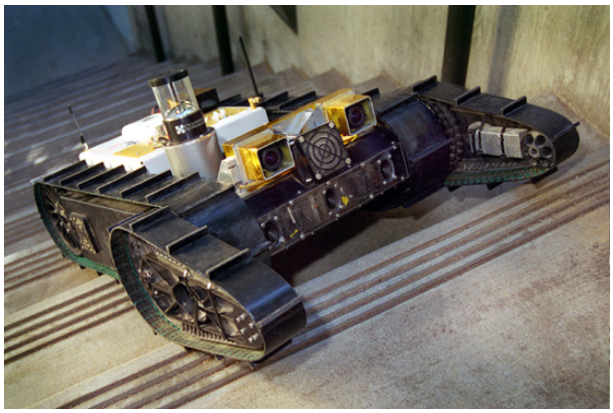

Figure 1. *The packbot vehicle.*

2. NAVIGATION SENSORS

The primary constraints for the navigation payload are accuracy, size, and power: the sensors must fit within the space and power budgets afforded by the chassis while delivering the resolution to reliably determine the position of the robot. Since the packbot is an autonomous platform, all perception, computation, and power resources are carried on board. The packbot is built upon the Urban II platform (shown in Figure 1) developed at iRobot Corporation. The chassis is approximately 60cm long, 50cm wide and 17cm tall with roughly 13,000cm^3 of contiguous payload space. A 20-cell NiCd battery pack provides a total energy storage of 120Wh. Power consumption with the robot standing still is approximately 75W, and the power required for driving varies with the terrain. The robot and all subsystems must be able to survive the shock of being thrown or dropped modest distances.

Autonomous path following and generation requires that both the leader and the follower have tight control over their respective positional accuracy. The accuracy of the sensor directly limits the ability to follow a path precisely. The accuracy limit and resolution is dictated by the terrain and by the follower's level of autonomy. If it is desired to have a follower blindly weave through a series of tightly spaced obstacles, i.e. trees, then the accuracy of the estimated position needs to be high - roughly half the width of the robot. On the other hand, if it is desired to have a reactive follower weave through the same obstacles, the accuracy requirements can be relaxed.

In order to meet the requirements of operating in such varied and unstructured environments, a combination of GPS and inertial sensors is used. GPS information is used in clear and unobstructed outdoor areas while inertial sensor data is used to augment GPS in areas where there is poor or no reception such as urban canyons, tree canopies, or indoors. However, there is no commercially available integrated package that can reasonably satisfy all the desires enumerated here.

2.1. GPS Receivers

Table 1 lists the commercially available GPS receivers that were considered. Positional accuracy is increased by an order of magnitude by using a differential base station to provide corrections. The real-time kinematic feature increases the resolution by another order of magnitude. It would be advantageous to have both of these features.

The NovAtel Millennium RT-2 was selected for the initial system and has performed well. We found that heat generation and large turn-on transients (∼5A) were a problem with this receiver. To solve these problems, we plan to migrate to a card that uses less power.

2.2. Inertial Navigation Sensors

Table 2 lists the commercially available integrated IMU packages that have been studied throughout this program. At the start of the program, no integrated system was available which would fit into the payload. Therefore a system was built from separate components.

A TCM2-50 compass/inclinometers from Precision Navigation provides heading and tilt. Three orthogonally mounted QRS11-200 rate gyros from Systron Donner and an EGCS3 three-axis accelerometer from Entran are used to measure angular rates and inertial forces. Careful consideration has to be given to the entire signal path from the sensor to the analog-to-digital converter since the analog output of the gyros and accelerometers are proportional to the rate of rotation and magnitude of acceleration. Initially, excessive noise in the signal caused the perceived drift rate to increase dramatically. Several steps were taken to reduce the noise including separating and isolating power supplies solely for the gyros and accelerometers, isolating the power and signal traces on the sensor and control board, shielding the signal traces, and taking special care of the routing of the signal traces to avoid cross-talk between the axes.

Since the beginning of the program, fully integrated MEMS inertial measurement units have become available. Their digital output eliminates the need for an analog-to-digital converter and increases noise immunity. These sensors are much smaller,

Manufacturer Model		BAE/CMC Superstar	BAE/CMC RT-Star	Motorola Oncore SL	NovAtel Millennium STD	NovAtel Millennium RT-2
Physical Characteristics						
Power	**W**	0.60	2.00	1.00	6.75	6.75
Length	**cm**	7.1	10.2	8	17.9	17.9
Width	**cm**	4.6	6.7	4	10	10
Depth	**cm**	1.3	1.4	1.2	1.5	1.5
Volume	**cc**	42.5	95.7	38.4	268.5	268.5
Mass	**g**	22	50	22	175	175
Operating Ranges						
Shock (endurance)	**g (1ms, 1/2 sine)**	500	500	200	400	400
Vibration (operating)	**g (20Hz-2kHz)**	5	5	7.7	4	4
Temperature	^oC	-30 to 75	-30 to 75	-40 to 85	-40 to 85	-40 to 85
Communication						
Comm. Method		TTL Level	TTL Level	TTL Level	RS-232	RS-232
Number of Ports		2	2	2	2	2
Update Rate	**Hz**	1	5	1	10	10
Performance Characteristics						
Velocity	m/s	515	515	515	515	515
Acceleration	m/s^2	39.2	39.2	39.2	58.9	58.9
Jerk	m/s^3	2	2	5	5	5
Number of Channels		12	12	8	24	24
Signals tracked		L1	L1	L1	L1/L2	L1/L2
Time to first fix (cold)	**s**	120	120	90	70	70
Time to first fix (warm)	**s**	45	45	45	60	60
Reacquisition time	**s**	3	3	1	5	5
Positional Accuracy (CEP)						
Stand-alone (w/o SA)	**m**	16	16	25	11	11
DGPS (w/o SA)	**m**	1	1	1	0.6	0.2
DGPS + RTK (w/o SA)	**m**	N/A	0.1	N/A	0.1	0.1
Time to convergence (DGPS, 1km)	**s**	N/A	200	N/A	200	180

Table 1. GPS Receiver Comparison.

Manufacturer Model		Packbot	Crossbow IMU400CA-100	Crossbow AHRS400CA-100	American GNC cmIMU	American GNC cmIMU+AHRS	BAE Systems SiIMU01
Degrees of Freedom		9DOF	6DOF	9DOF	6DOF	9DOF	6DOF
Method		3G, 3A, 3Mag	3G, 3A	3G, 3A, 3Mag	MEMS	MEMS + 3Mag	MEMS
Physical Characteristics							
Power	**W**	2.80	3.00	4.20	1.50	3.00	2.50
Length	**cm**	7.62	7.62	7.62	2.5	3.66	7.49
Width	**cm**	5.08	9.53	9.53	2.5	3.66	
Height	**cm**	6.35	8.13	10.41	2.5	3.66	4.55
Volume	**cc**	245.8	590.4	756.0	15.6	48.9	200.5
Mass	**g**	450	590	640	85	85	255
Operating Ranges							
Shock (endurance)	**g (1ms, 1/2 sine)**	200	1000	1000	500	500	1000
Vibration (operating)	**g (20Hz-2kHz)**	8	10	10	7	7	20
Temperature	^oC	-20 to 70	-40 to 70	-40 to 70	-40 to 80	-40 to 80	-46 to 85
Communication							
Comm. Method		Analog	RS-232	RS-232	I^2C	I^2C	RS-422/485
Update Rate	**Hz**	16	100	60	2000	500	200
Data Latency	**ms**	N/R	3.5	25	0.75	0.75	1.25
Time to first fix	**s**	1	1	1	0.2	0.2	0.2
Time to full accuracy	**s**	N/A	1	60	90	90	60
Gyros							
Bandwidth	**Hz**	60	10	10	500	500	75
Range	$^\circ/sec$	200	100	100	150	150	150
Resolution	$^\circ/sec$	0.01	0.025	0.025	0.01	0.01	0.01
Non-linearity	**%FS**	1	0.3	0.3	0.1	0.1	0.01
Random Walk	$^\circ/h^{1/2}$ **(ARW)**	N/R	0.85	0.85	0.1	0.1	0.2
Drift	$^\circ/h$	17	20	20	17	17	2
Accelerometers							
Bandwidth	**Hz**	280	100	100	500	500	75
Range	$\pm g$	5	2	2	5	5	15
Resolution	**mg**	33	5	5	0.01	0.01	0.03
Non-linearity	**%FS**	1	1	1	0.05	0.05	0.004
Random Walk	$m/s/h^{1/2}$ **(VRW)**	0.15	0.15	N/R	N/R	0.12	
Bias	**mg**	24.9	8.5	8.5	0.2	0.2	2
AHRS							
Pitch Range	$\pm^\circ$	50	N/A	90	N/A	90	N/A
Roll Range	$\pm^\circ$	50	N/A	180	N/A	180	N/A
Heading Range	$^\circ$	360	N/A	360	N/A	360	N/A
Angular Resolution	$^\circ$	0.1	N/A	0.1	N/A	0.05	N/A
Static Accuracy	$^\circ$	0.4	N/A	1.5	N/A	N/R	N/A
Dynamic Accuracy	$^\circ$	N/R	N/A	3	N/A	0.05	N/A

Table 2. Inertial Navigation Sensor Comparison.

requiring a half to a quarter of the volume of conventional integrated systems. Additionally, these units are being packaged with magnetometers so that a complete Attitude Heading Reference System (AHRS) solution is possible.

3. POSITION ESTIMATION

Precise localization is one of the main requirements for the task of autonomous path following. The packbot mobile robot is equipped with differential GPS (DGPS) that provides position estimates with 2-20cm uncertainty under favorable conditions. These uncertainties can become much higher when operating near buildings or trees, which occlude satellite signals making GPS navigation unreliable. During GPS dropouts, the signals from the inertial sensors, compass/inclinometers, and motor encoders have to be appropriately combined so as to determine the location of the robot until the next GPS update. By integrating accurate estimates of its linear and rotational velocity the packbot could potentially track its pose for a long period of time. As the robot turns using skid steering the inherent slippage makes the estimates based on the motor encoder signals untrustworthy,

especially the ones regarding rotational velocity. Since even small errors in orientation can cause large errors in position, we focus on deriving precise heading estimates.

The three Systron Donner gyroscopes set in an orthogonal configuration are used for this purpose. Appropriate integration of their signals (angular rates) provides estimates of the roll, pitch and yaw angles that determine the attitude of the vehicle. A common difficulty found in all approaches that rely on gyros for attitude estimation is the low frequency noise component (also referred to as bias or drift) that violates the white noise assumption required for standard Kalman filtering. Inclusion of the gyro noise model in a Kalman filter by suitably augmenting the state vector has the potential to provide estimates of the sensor bias when the observability requirement is satisfied. The system becomes observable when absolute orientation measurements are available. In the case of the packbot, this information is provided by the TCM2 compass/inclinometers module. The inclinometers measure the attitude of the robot with respect to the horizontal plane while the compass provides a measurement of the direction of the vehicle compared to the magnetic north. Before describing in detail the Kalman filter implementation for estimating the orientation of the vehicle, we first examine the model employed to capture the effect of the gyroscope noise.

3.1. Noise Model for the Systron Donner Quartz Gyro

In the information provided in[4] it is obvious that the Systron Donner gyro does not have a stable bias. From page 1-4: *"Low Rate Application - These gyros showed reasonable performance for rate scale factor stability but would not be useful for applications where bias stability was of high importance to meet mission requirements. The bias changed significantly as the input rate was changing making predictable bias compensation very difficult".*

Long term bias stability data were gathered to create a stochastic model useful for attitude estimator performance prediction. This model assumes that the gyro noise is composed of 3 elements, namely: rate noise $n_r(t)$ (additive white noise), rate flicker noise $n_f(t)$ (generated when white noise passes through a filter with transfer function $1/\sqrt{s}$) and rate random walk $n_w(t)$ (generated when white noise passes through a filter with transfer function $1/s$). The Power Spectral Density (PSD) of the gyro noise was measured experimentally and the logarithmic plots of the PSD with respect to frequency were used to fit the described model. The intensities calculated (ignoring the flicker noise) were: $\sigma_r = \sqrt{N_r} = 0.009\ (^\circ/sec)/\sqrt{Hz}$ and $\sigma_w = \sqrt{N_w} = 0.0005012\ (^\circ/sec)\sqrt{Hz}$.

Based on this model[5] the angular velocity $\omega = \dot{\theta}$ is related to the gyro output ω_m according to the equation:

$$\dot{\theta} = \omega_m - b - n_r, \qquad E[n_r(t)] = 0, \qquad E[n_r(t)n_r'(t')] = N_r\delta(t-t') \tag{1}$$

where b is the drift-rate bias and n_r is the drift-rate noise assumed to be a Gaussian white-noise process. The drift-rate bias is not a static quantity but is driven by a second Gaussian white-noise process, the gyro drift-rate ramp noise:

$$\dot{b} = n_w, \qquad E[n_w(t)] = 0, \qquad E[n_w(t)n_w'(t')] = N_w\delta(t-t') \tag{2}$$

These two noise processes are assumed to be uncorrelated ($E[n_w(t)n_r'(t')] = 0$).

3.2. TCM2-50 Electronic Compass/Inclinometers Characterization

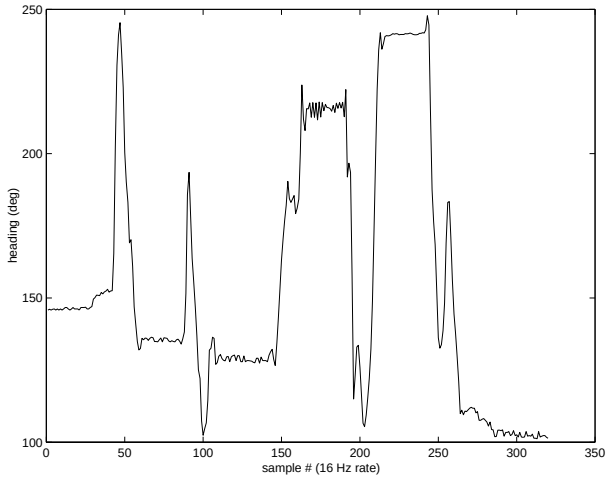

Figure 2. *Effect of track movement on compass heading.*

As mentioned the packbot is equipped with a TCM2 module that contains a compass and two inclinometers. The compass is comprised of 3 orthogonal magnetometers that measure the local intensity of the magnetic field of the earth. This information combined with the inclinometers' tilt angles - determined by the effect of the local vector of gravity on the contained viscous fluid - provides an absolute measurement of the attitude of the vehicle at a rate of 16Hz. The expected accuracy is $\pm 1.5^\circ$ for

heading and $\pm 0.4^\circ$ for tilt. These measurements are processed by the Kalman filter in order to estimate the gyroscopes' biases and reduce their influence on the orientation estimates.

Since the compass is affected by local magnetic fields present on the robot, the TCM2 module has to be calibrated in order to compensate for static fields. However, such procedure cannot deal with the dynamic fields produced by the metallic belts inside the tracks of the robot. Experimental testing of the compass while manually rotating the tracks has shown variances of over 130° (Figure 2). Now, nylon belted tracks are used whenever possible in order to avoid this problem.

When the vehicle is in motion the angles recorded by the inclinometers depend on both the gravitational vector and the accelerations due to the interaction with the ground. These measurements do not reflect the actual pose of the robot and for this reason, absolute orientation measurements are processed by the filter only when the robot is stopped.

3.3. Kalman filter based attitude estimation

3.3.1. Dynamic Model Replacement

In our implementation of a Kalman filter observer which estimates the orientation of the robot, we employed sensor modeling instead of dynamic modeling. The reasoning behind this is as follows:

i) The most elementary reason is that every time there is a modification of the robot (i.e. a mass changes, or a part is relocated, or dimensions are altered) the dynamic modeling would have to be redone. Since the produced estimator is tailored for a specific structure, a slightly different vehicle would require a new estimator.

ii) A more practical reason is that dynamic modeling would require a very large number of states. An estimator has to be practical as far as its computational needs are concerned. The size of the estimated state can have large computational demands with very little gain in precision.

iii) Dynamic modeling and the added complexity caused do not always produce the expected results. One example is an attempt by Lefferts and Markley[6] to model the attitude dynamics of the NIMBUS-6 spacecraft, which indicated that dynamic modeling with elaborate torque models could still not give acceptable attitude determination accuracy. For this reason most attitude estimation applications in the aerospace domain use gyros in a dynamic model replacement mode.[7]

iv) The modeling of a mobile robot moving on rough terrain is particularly complicated because of the interaction between the wheels of the vehicle and the ground. This interaction depends on many parameters which are difficult to measure and requires simplifying assumptions to be made (the point contact assumption is frequently used). The modeling of the vehicle-terrain dynamic effects (i.e. wheel impact, slippage, and sinkage) require prior knowledge of the ground parameters (i.e. friction coefficients as a function of wheel slippage ratio, soil shear strength, elastic modulus, etc). In addition, lateral slippage is not readily observable and cannot easily be accounted for in the dynamic model. Precise modeling of the motors is also required to obtain the real values of the torques acting on each of the wheels. Since the applicability of Kalman filtering techniques rests on the availability of an accurate model, an inaccurate dynamic model can cause the attitude estimate to drift away from its real value.

3.3.2. Indirect versus Direct Kalman filter

A key aspect of our implementation of a Kalman filter in conjunction with the inertial navigation systems (INS) is the use of the indirect instead of the direct form. These forms of the filter are also referred to as the error state and the total state formulation respectively.[8] As the name indicates, in the total state (direct) formulation, states such as orientation are amongst the variables in the filter, and the measurements are IMU outputs (e.g. from gyros) and external source signals (e.g. from compass/inclinometers). In the error state (indirect) formulation, the *errors* in orientation are amongst the estimated variables, and each measurement presented to the filter is the *difference* between the IMU based estimates and the external source data.

There are some serious drawbacks inherent to the direct realization of the Kalman filter. Being in the INS loop and using the total state representation, the filter has to maintain explicit, accurate awareness of the vehicle's angular motion and at the same time attempt to suppress noisy and erroneous data. Sampled data require a sampling rate of at least twice the highest frequency signal (in practice a factor of 5-10 is used) for adequate reconstruction of the continuous time system behavior. The filter would have to perform all the required computations within this short sampling period. Moreover, in most cases, the estimation algorithm is allocated only a small portion of the processor's clock cycles.[8] Frequently, it runs in the background at a lower priority than more critical algorithms, such as real-time vision, obstacle avoidance and fault detection.

In addition, the dynamics involved in the total state description of the filter include high frequency components and are well described only by a non-linear model. The development of a Kalman filter is predicated upon an adequate linear system model but such total state model does not exist.

Another disadvantage of the direct filter design is that if the filter fails (as by a temporary computer failure) the entire navigation algorithm will fail and the IMU would be useless without the filter. From the reliability point of view it would be desirable to provide an emergency degraded performance mode in such a case of failure.

The Indirect (error state) Kalman filter estimates the errors in the navigation and attitude information using the difference between the INS and external sources of data (e.g. compass/inclinometers). The INS itself is able to follow the high frequency motions of the vehicle very accurately, and there is no need to model these dynamics explicitly in the filter. Instead, the Indirect filter is based on a set of inertial system error propagation equations which are low frequency and adequately represented as linear. Because the filter is out of the INS loop and is based on low frequency dynamics, its sampling rate can be much lower

than that of the direct filter. In fact an effective Indirect filter can be developed with a sample period (of the external source) on the order of minutes.[8] This is very practical with respect to the amount of computer resources required. For these reasons, the error state formulation is used in essentially all terrestrial aided inertial navigation systems.[9]

3.3.3. Quaternions in Attitude Representation

The three-parameter Euler angle representation has been used in most applications of the Kalman filter in robot localization.[10,11] However the kinematic equations for Euler angles involve non-linear and computationally expensive trigonometric functions. The computational cost using quaternions is less than using Euler angles It is also more compact because only four parameters, rather than nine, are needed. Furthermore in the Euler angle representation the angles become undefined for some rotations (the gimbal lock situation) which causes problems in Kalman filtering applications. Amongst all the representations for finite rotations, only those of four parameters behave well for arbitrary rotations. The reason is that a non-singular mapping between parameters and their corresponding rotational transformation matrix requires a set of at least four parameters.[12]

Quaternions provide an elegant and convenient parameterization of the attitude. A unit quaternion is a global non-singular four parameter representation also known as Euler symmetric parameters. Physical quantities pertaining to the motion of rotation such as angular displacement, velocity, acceleration and momentum are derived in terms of Euler parameters in a simple manner. Manipulating those equations is much easier using quaternion algebra.[13] The generalized commutative properties of quaternion multiplication are very useful when combining kinematic equations derived from successive rotations in space. The attitude matrix computed from a quaternion (as a quadratic function) is orthogonal when the sum of squares of the quaternion components is unity. If propagation errors result in a violation of this constraint the quaternion can be renormalized by dividing its components by the (scalar) square root of the sum of their squares.

3.3.4. Attitude kinematics and error state equations

The physical counterparts of quaternions are the rotational axis $\hat{n}$ and the rotational angle θ that are used in the Euler theorem regarding finite rotations. Taking the vector part of a quaternion and normalizing it, we can find the rotational axis right away, and from the last parameter we can obtain the angle of rotation.[14] Following the notation in[15] a unit quaternion is defined as:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{3}$$

with the constraint

$$q^T q = 1 \tag{4}$$

where

$$q_1 = n_x sin(\theta/2), \qquad q_2 = n_y sin(\theta/2), \qquad q_3 = n_z sin(\theta/2), \qquad q_4 = cos(\theta/2) \tag{5}$$

and $\hat{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$ is the unit vector of the axis of rotation and θ is the angle of rotation.

The rate of change of the quaternion with respect to time is given by:

$$\frac{d}{dt}q(t) = \frac{1}{2}\Omega(\vec{\omega}(t))q(t), \qquad \Omega(\vec{\omega}) = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \tag{6}$$

where $\vec{\omega} = \dot{\vec{\theta}}$ is the rotational velocity vector. At this point we present an approximate body-referenced representation of the error state vector. The error state includes the bias error and the quaternion error. The bias error is defined as the difference between the true and estimated bias.

$$\Delta\vec{b} = \vec{b}_{true} - \vec{b}_i \tag{7}$$

The quaternion error here is not the arithmetic difference between the true and estimated (as it is for the bias error) but it is expressed as the quaternion which must be composed with the estimated quaternion in order to obtain the true quaternion. That is:

$$\delta q = q_{true} \otimes q_i^{-1} \Leftrightarrow q_{true} = \delta q \otimes q_i \tag{8}$$

The advantage of this representation is that since the incremental quaternion corresponds very closely to a small rotation, the fourth component will be close to unity and thus the attitude information of interest is contained in the three vector component $\delta\vec{q}$ where

$$\delta q \simeq \begin{bmatrix} \delta\vec{q} \\ 1 \end{bmatrix} \tag{9}$$

Starting from equations:

$$\frac{d}{dt}q_{true} = \frac{1}{2}\Omega(\dot{\vec{\theta}}_{true})q_{true} \tag{10}$$

and

$$\frac{d}{dt}q_i = \frac{1}{2}\Omega(\vec{\theta}_i)q_i \tag{11}$$

where $\vec{\theta}_{true}$ is the true rate of change of the attitude and $\vec{\theta}_i$ is the estimated rate from the measurements provided by the gyros, it can be shown[16] that

$$\frac{d}{dt}\delta\vec{q} = [[\ \vec{\omega}_m\ ]]\,\delta\vec{q} - \frac{1}{2}(\Delta\vec{b} + \vec{n}_r)\quad,\quad \frac{d}{dt}\delta q_4 = 0 \tag{12}$$

with

$$[[\ \vec{\omega}_m\ ]] = \begin{bmatrix} 0 & \omega_3 & -\omega_2 \\ -\omega_3 & 0 & \omega_1 \\ \omega_2 & -\omega_1 & 0 \end{bmatrix} \tag{13}$$

Using the infinitesimal angle assumption in Equation (5), $\delta\vec{q}$ can be written as

$$\delta\vec{q} = \frac{1}{2}\delta\vec{\theta} \tag{14}$$

and thus Equation (12) can be rewritten as

$$\frac{d}{dt}\delta\vec{\theta} = [[\ \vec{\omega}_m\ ]]\,\delta\vec{\theta} - (\Delta\vec{b} + \vec{n}_r) \tag{15}$$

Differentiating Equation (7) and making the same assumptions for the true and estimated bias as in the previous section (Equations (1) and (2)), the bias error dynamic equation can be expressed as

$$\frac{d}{dt}\Delta\vec{b} = \vec{n}_w \tag{16}$$

Combining Equations (15) and (16) we can describe the error state equation as

$$\frac{d}{dt}\begin{bmatrix} \delta\vec{\theta} \\ \Delta\vec{b} \end{bmatrix} = \begin{bmatrix} [[\ \vec{\omega}_m\ ]] & -I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix}\begin{bmatrix} \delta\vec{\theta} \\ \Delta\vec{b} \end{bmatrix} + \begin{bmatrix} -I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \end{bmatrix}\begin{bmatrix} \vec{n}_r \\ \vec{n}_w \end{bmatrix} \tag{17}$$

or in a more compact form

$$\frac{d}{dt}\Delta x = F\Delta x + Gn \tag{18}$$

This last equation describes the system model employed in the current Kalman filter implementation.[16] This estimator combines the gyroscopes angular rates with the absolute orientation measurements from the compass/inclinometers in order to estimate both the attitude of the vehicle and the gyro biases. As shown in[17] this estimator acts as a high pass filter on the gyro signals filtering out the low frequency noise component (bias) while weighing more their contribution during high frequency motion when the compass/inclinometers are susceptible to disturbances. If absolute orientation measurements are available continuously the filter is capable of continuously tracking the gyro biases (Figure 3). In our case though, since the robot uses the TCM2 sensor only when stopped, the filter updates its estimate of the bias (Figure 4) only intermittently based on its effect on the attitude estimates during the previous interval of motion. The resulting attitude estimates (Figure 5) are then combined with the translational velocity measurements (from the encoders) to provide position estimates inbetween GPS updates.* Although GPS data are nominally available at 10Hz, during dropouts the robot has to rely on the estimates provided by the filter in order to follow a given path.

4. SYSTEM ARCHITECTURE AND LEADER FOLLOWER CONTROL

The system architecture used to control the packbot is designed to allow multiple behaviors to command the robot simultaneously. The driving commands from these behaviors are arbitrated upon, and from them a final command is composed. This allows several behaviors using multiple sensors and imagers to work together effectively to carry out the commanded mission goals (see Figure 6). A path following controller based on "carrot following" has been implemented and integrated into the packbot system. This algorithm takes in a set of points that form the path to be followed and runs a combination of a pure pursuit and a proportional-integral-derivative control scheme to direct the robot along the path.

*We are in the process of enhancing the current Kalman filter implementation so as to fuse data from the accelerometers and the GPS and provide improved position estimates.

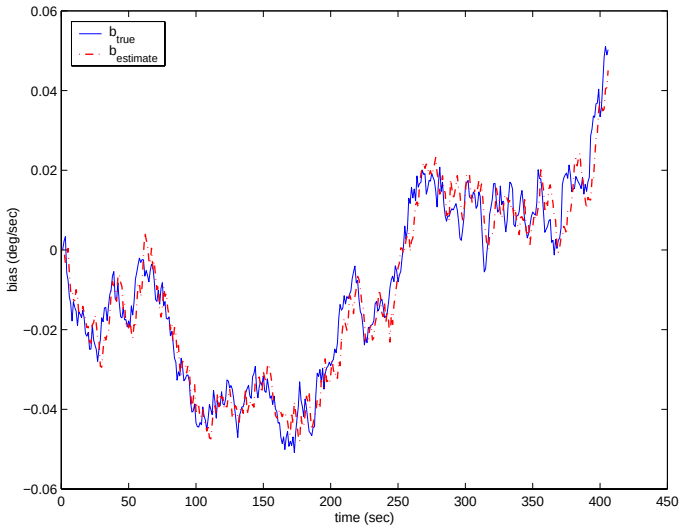

Figure 3. *Bias Estimation (simulation results): The solid line represents the true value of the gyro bias. The dotted line is the estimate of the bias from the Indirect Kalman filter. Though the estimate follows the true value it is obvious that there is a lag. This is because the absolute orientation sensor does not measure the bias directly. It only measures its effect on the orientation error.*

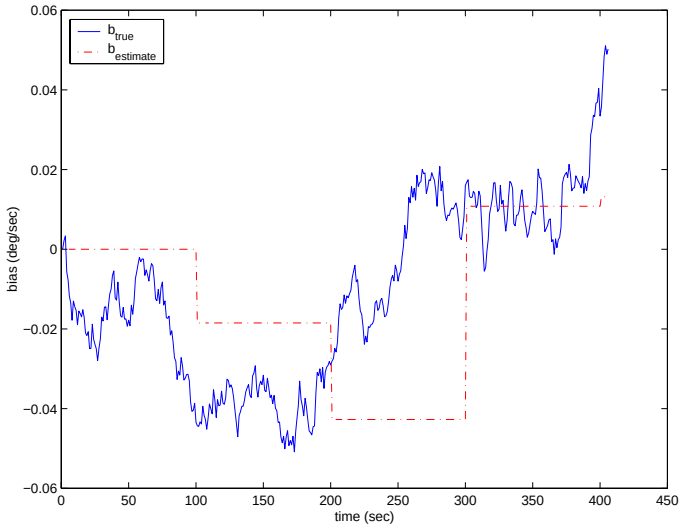

Figure 4. *Bias Estimation (simulation results): The flat parts of the estimate depict the constant bias assumption in the integrator. The sharp step changes occur when absolute orientation measurements become available (every 100sec).*

4.1. System Architecture

The navigation sensors are managed by device drivers which pass data through a software message queue to a single software task which carries out the necessary calculations. This task runs the Kalman filter and position estimation algorithms after each piece of sensor data comes in. It then updates the current state of the robot in a shared memory space where other tasks can access it. Currently the GPS is used to determine the robot's position both for recording and following paths. When the robot is indoors, or when it drives into GPS-dropout areas, the position estimation is calculated with a simple interpolation using the Kalman-filtered heading and a raw odometry estimate from the wheel encoders.

The path recording and path following code, as well as other software tasks, can access the latest robot position and orientation estimates at variable rates and make decisions accordingly. A software task monitors the robot's current position and records a 3-D point after a certain constant offset has been passed thereby forming the robot's trail from successive points. The robot's trail is accessed, downloaded, and then edited using the Operator Control Unit, a GUI running on a laptop that is used to control and test the packbot. The modified or whole trail is sent to another robot or back to the same packbot, which accepts the trail, passes it to the path following module, and begins driving with a "go" command.

Another algorithm that has been tested and used on the packbot is the Obstacle Detection and Obstacle Avoidance (ODOA) behavior. It uses a stereo camera pair to build a range map and obstacle map of the area in front of the robot. ODOA is

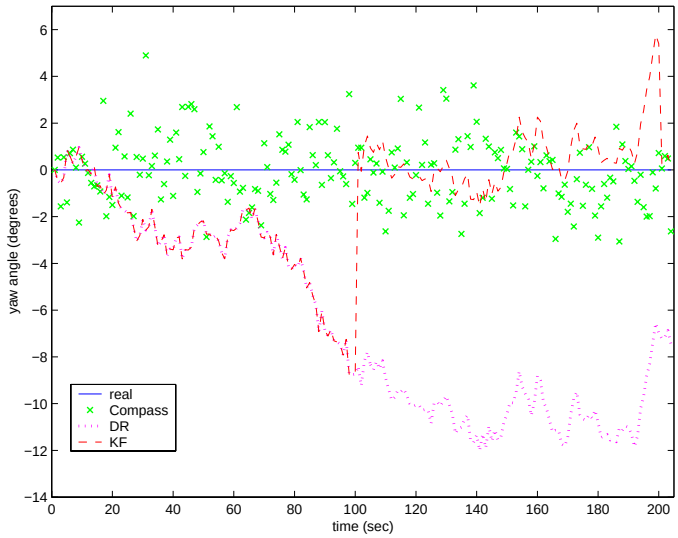

Figure 5. *Attitude Estimation (simulation results): The solid line represents the actual orientation (yaw) of the robot, the dotted line shows the dead-reckoned yaw estimates obtained by simply integrating the gyro signals, and the dashed line corresponds to the Kalman filter estimates when compass measurements are provided to the filter intermittently (t=100, 200 sec). Even though all the compass measurements during this interval are displayed here, only two of them were processed for the orientation updates of the filter. The two sharp corrections of the yaw estimates correspond to the time instants that these measurements became available to the filter.*

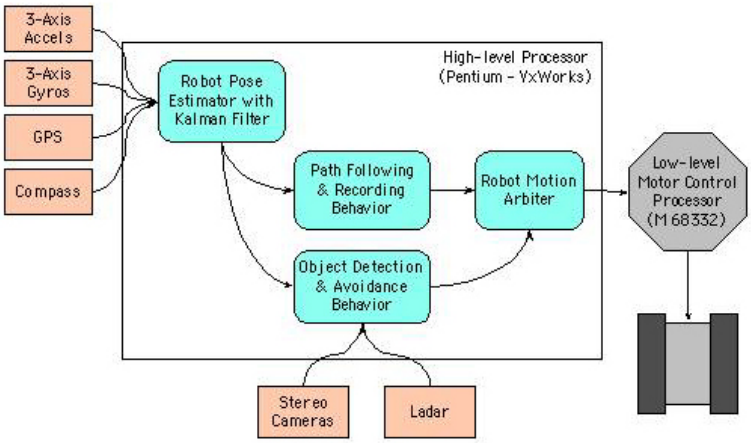

Figure 6. *System Architecture.*

also capable of using an on-board ladar scanner to detect obstacles and avoid them. Both the path following and the ODOA behaviors output driving commands in the format of votes which are sent to the system motion arbiter.

The arbiter is a software task that takes input votes from two or more behaviors and composes them into one driving command. The input votes are a set of twenty three forward arcs of different curvature, twenty three backward arcs of different curvature, and two turn-in-place votes. Each of the arcs in each vote sent to the arbiter has a robot speed and "desire" weight. The behavior sending the vote indicates the preference of driving direction with different desire weights along each arc and completely prohibits an arc by assigning it zero. The arbiter determines the chosen driving direction and speed by averaging the weights of the arcs and using the minimum speed of the largest weighted arc. In this way, protective behaviors can modify the motion of the robot as directed by goal-seeking behaviors thereby, providing autonomous self-protection for the packbot.

4.2. Path following control algorithm

Path-following steering control is based on a carrot-following approach. A set of subgoals is transmitted to the follower vehicle. The follower vehicle uses the last two subgoals to extend the path sequence backwards, as shown in Figure 7. The follower vehicle then uses this extended path segment to locate a carrot position that lies a lookahead distance L away from the vehicle center. The carrot position is updated every cycle by the path following algorithm. At any given instant, the follower assigns the largest weight to the arc that passes closest to the carrot.

The carrot is located by finding the intersection of a circle (centered at the vehicle center with radius L) and the extended

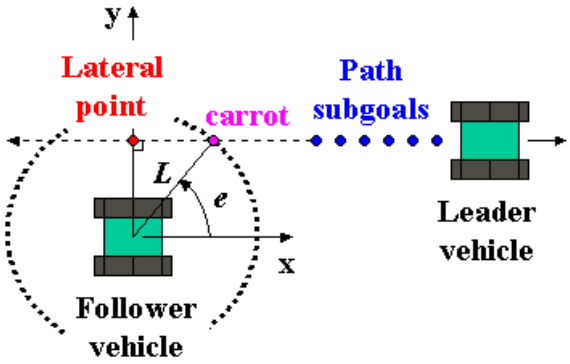

Figure 7. *Carrot Following Approach.*

path segment. If there are two intersections, the one beyond the lateral position is chosen. If there is no intersection, the lateral position is chosen as the carrot. The steering curvature that will move the vehicle center directly over the carrot position is chosen as the commanded curvature.

4.3. Pure Pursuit Controller

The authors have experimented with two controllers that produce command curvatures: pure pursuit and proportional-integral-derivative (PID) control. Pure pursuit has been widely used as a steering controller for autonomous vehicles.[18,19] Amidi and Thorpe[20] compared pure pursuit with a quintic polynomial fit method and a classic control theory approach. Ollero and Heredia[21] analyzed the stability of the pure pursuit algorithm for path following at constant speed (3, 6, and 9 m/s) for straight and constant curvature path sections, estimating the time lag for computing, communications, and actuator delay. Kelly[19] has described an adaptive pure pursuit controller, allowing the look-ahead gain to increase as a function of the lateral path error. Rankin[22] has evaluated a pure pursuit controller, a PI controller, and a weighted pure pursuit/PI controller.

The controlling equation for pure pursuit is shown in Equation (19). $k_{pure\ pursuit}$ is the command curvature and y_{carrot} is the y coordinate of the carrot position as measured in the vehicle coordinate system. Pure pursuit is a proportional controller, where, y_{carrot} represents the current error and $(2/L^2)$ represents the proportional gain. The lookahead distance L should be appropriate for the mission speed.

$$k_{pure\ pursuit} = \left[\frac{2}{L^2}\right] y_{carrot} \tag{19}$$

4.4. PID Controller

The second method of steering control that was implemented is a proportional-integral-derivative (PID) controller, as applied to the error in the vehicle's heading. The idealized equation of a PID controller is shown in Equation (20). PID control contains a term that is proportional to the error, one that is proportional to the integral of the error, and one that is proportional to the derivative of the error.[23]

$$k_{PID}(t) = G\left[e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau + T_D\frac{de(t)}{dt}\right] \tag{20}$$

The integral term acts as a spring (in a spring-mass-damper system) in that it eliminates steady state error. The derivative term acts as a damper. The parameters that are characteristic to the system are the proportional gain G, the integral time T_I, the derivative time T_D, and the sampling time T. For small sample times T, the idealized equation can be written as a nonrecursive difference equation, as shown in Equation (21).

$$k_{PID,n} = G\left[e_n + \frac{T}{T_I}\sum_{i=0}^{n-1} e_i + \frac{T_D}{T}(e_n - e_{n-1})\right] \tag{21}$$

The heading error e_n is calculated by finding the orientation of the vector from the vehicle's control point to the target position with respect to the axis in the direction of the vehicle's current heading. A positive error results in a left turn and a negative error results in a right turn. Equation (21) can be rewritten as the difference equation

$$k_{PID,n} = k_{PID,n-1} + q_0 e_n + q_1 e_{n-1} + q_2 e_{n-2} \tag{22}$$

where,

$$q_0 = G\left[1 + \frac{T_D}{T}\right] = g_P + \frac{g_D}{T}$$

$$q_1 = -G\left[1 + 2\frac{T_D}{T} - \frac{T}{T_I}\right] = -g_P - 2\frac{g_D}{T} + g_I T$$

$$q_2 = G\left[\frac{T_D}{T}\right] = \frac{g_D}{T}$$

The PID controller works to force the heading error to zero so that the vehicle is always pointed towards the current carrot position. The parameters g_P, g_I, and g_D are respectively proportional, integral and derivative gains. When $g_D = 0$, the PID controller is reduced to a proportional-integral (PI) controller.

4.5. Combined Pure Pursuit/PID Control

Both the pure pursuit and PID methods of steering control have advantages and disadvantages. The pure pursuit controller is easy to tune and performs well when the follower vehicle is started on or near the extended path segment. If the lateral path error is large, however, this method can become unstable. Stability can be improved by using an adaptive pure pursuit controller. With the adaptive version, the look-ahead distance is a function of the lateral path error. The adaptive controller, however, can cause a significant portion of the path segment (traversed by the leader vehicle) to be ignored by the follower vehicle.

The PID method is stable (when adequately tuned) over the range of heading errors. This includes the scenario where there is a large lateral path error. This controller does, however, cause an inherent lateral path error when traveling around curves. In an attempt to combine the desirable features of both the standard pure pursuit and PID controllers into a single controller, the output of each controller is averaged, as shown in Equation (23). The arc that is closest to this curvature is then assigned the largest weight.

$$k_{command} = \frac{k_{pure\ pursuit} + k_{PID}}{2} \tag{23}$$

5. EXPERIMENTAL RESULTS

5.1. Indoor Testing

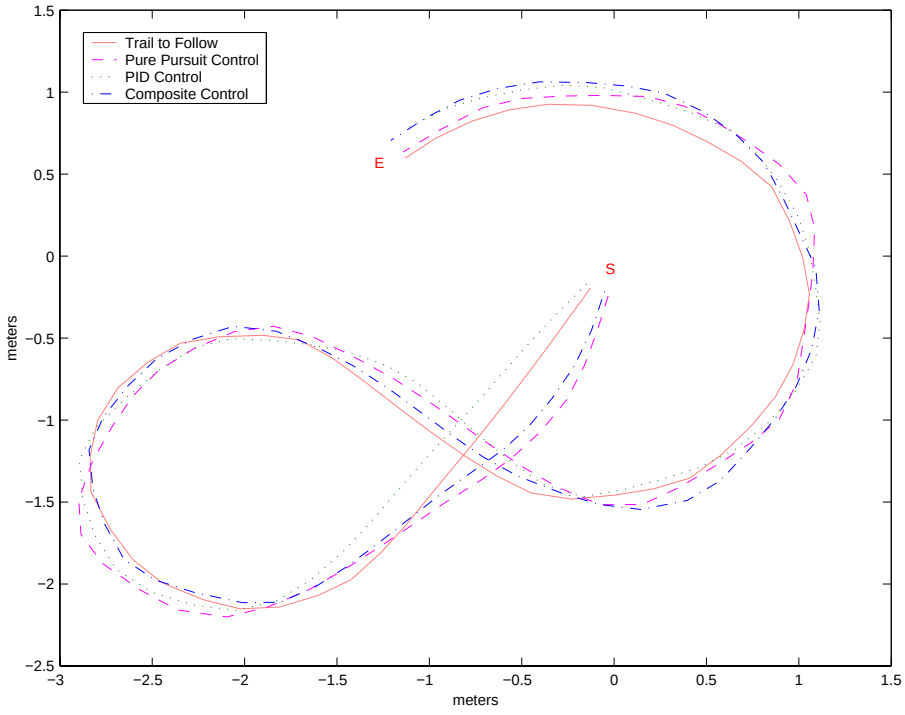

Figure 8. *Indoor Testings.*

The path following algorithm is currently being run at a rate of 10 Hz which is the same rate at which the position estimate is updated using either odometry or GPS data. Due to the reporting speed of the inertial navigation sensors, the Kalman filter updates the heading at over 256 Hz. During all tests the robot traversed the paths at a speed of 50cm/sec. To provide enough detail to describe the path without accumulating an unnecessary number of points, an interval of approximately 20 cm between points was used to record the robot's trail.

Initially, several indoor runs were used to test the follower algorithm and tune its parameters. Figure 8 shows a path recorded by the packbot and the three separate path-following runs carried out by the same robot using the different control methods (pure pursuit, PID, and the combination of the two). In the indoor tests no GPS data could be received so all position estimates used the Kalman filter heading and odometry from the wheel encoders. This also means that only relative robot coordinates were known so even though the start and finish of the runs are at the same points in the graph, in actuality the robot was possibly starting in very different locations.

The pure pursuit method tended to oscillate over the path being followed more than the PID but it also managed to stay closer to the path in general. The PID control kept the robot's heading going in a parallel direction with the path but maintained a slight offset for longer period of time than the pure pursuit control. Currently further work is being done to tune the PID controller to better suit the packbot system. The average of the controller error was used as a numerical measure of the performance of each algorithm across various runs. The simple combination of the two controllers consistently reported the lowest controller-error averages.

5.2. Outdoor Results

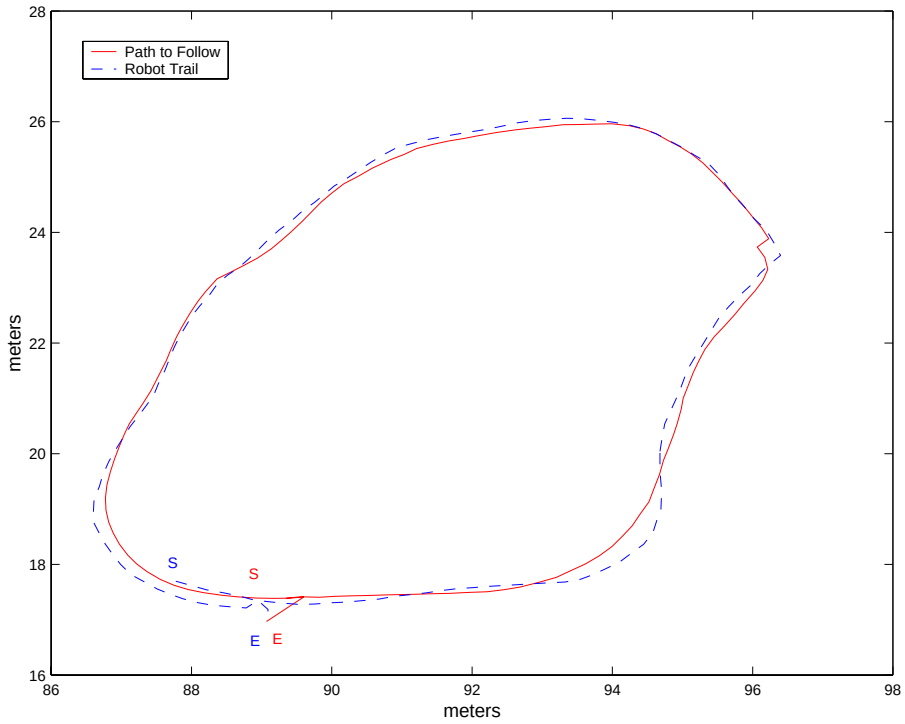

Figure 9. *Outdoor following.*

In outdoor runs the packbot again was used to trace a path and then was given its own trail to follow. During these tests the Kalman filter provided the robot's heading and the GPS gave position data. While in motion the compass/inclinometers noise is higher and thus the Kalman filter relies more on the gyro package for heading approximation. After some initial experimentation several metal objects embedded in the concrete of the testing ground were found to be causing the magnetometer to provide incorrect readings to the Kalman filter which in turn caused an incorrect initial heading. This heading offset caused a corresponding constant offset of the robot from the path it was following. These metal objects were then avoided in choosing starting positions for the robot.

In Figures 10 and 11 a GPS dropout is indicated. When driving under the obstacle, a small SUV, all GPS data was cut off and the robot position estimator had to rely on the Kalman filter heading and wheel odometry exclusively. Using this method the recorded trail was interpolated through the GPS dropout smoothly and the packbot was able to follow the path up to, under, and past the obstacle without any noticeable problems. The GPS data converged approximately 10-14secs after leaving the dropout area at which point the position estimator switched back to using it to report the packbot's location.

Figure 10 also shows a few sudden changes in the coordinates of the path followed due to unfiltered GPS jumps. As directed by the follower behavior, the packbot traversed these segmented pieces of the path smoothly.

6. SUMMARY AND EXTENSIONS

In summary, we have integrated three-axis gyros and accelerometers, a compass/inclinometer package, track odometry, differential GPS, and an indirect, error-state Kalman filter into a sensor system for small robot position estimation. We summarized

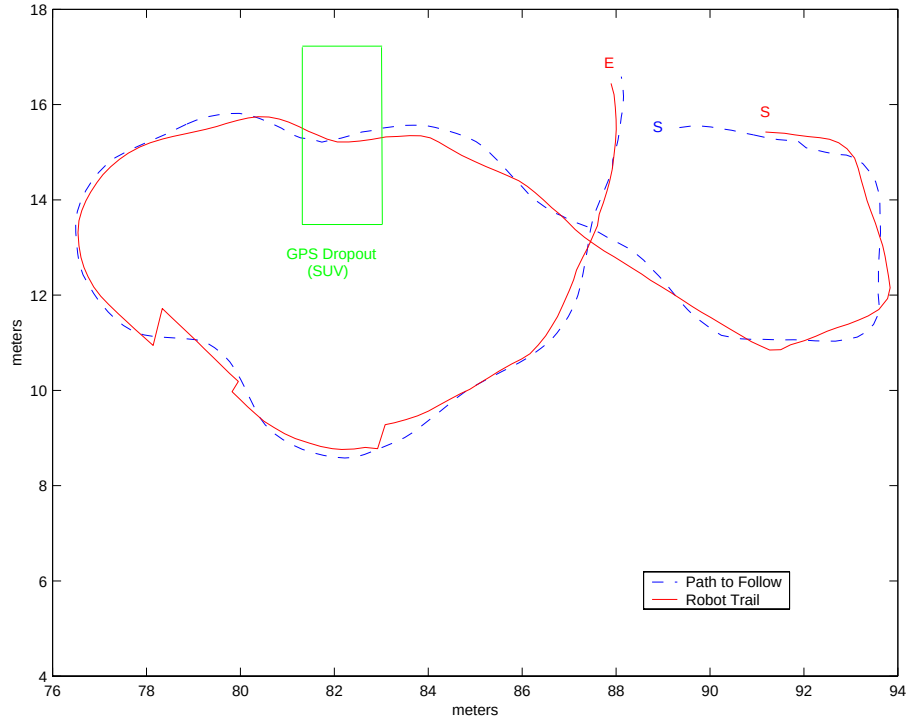

Figure 10. *Outdoor following with GPS skips and dropout.*

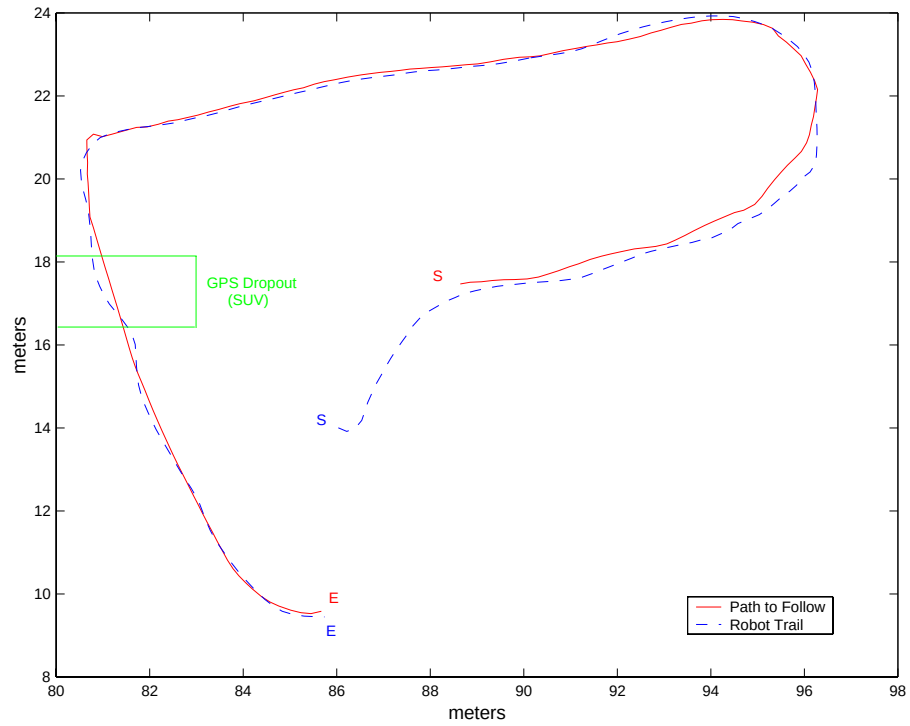

Figure 11. *Outdoor following with GPS dropout.*

problems that arose with these sensors and the solutions we found. Our sensor survey included integrated IMUs that will soon be on the market. These are smaller than the combination of sensors used here and have advertised performance specifications that are at least as good as the sensors we use now; these IMUs are attractive for future systems.

In the leader/follower behavior, we implemented three versions of the path-following controller: pure pursuit, PI, and an average of the two. Indoor and outdoor experiments over up to roughly 40 meters showed good performance for all three controllers, with maximum path deviations on the order of 50 cm. This included segments where the path went under vehicles that caused GPS dropouts and paths with sharp kinks due to GPS outliers. The averaging controller showed better error performance

than either pure pursuit or PI alone, but we have not done extensive characterization testing to date.

In future robot systems and experiments, position estimation performance may be worse than reported here due to larger areas of GPS dropout or the use of smaller, lower cost GPS units with poorer precision. We are currently extending our system to include optimal path smoothing to address GPS dropouts and to include special-purpose landmark recognition for path-following through constricted areas of known types, in particular culverts and doorways. We are also pursuing more general outdoor mapping and landmark recognition algorithms to further reduce the reliance on GPS.

REFERENCES

1. L. Matthies, Y. Xiong, R. Hogg, D. Zhu, A. Rankin, B. Kennedy, M. Hebert, R. Maclachlan, C. Won, T. Frost, G. Sukhatme, M. McHenry, and S. Goldberg, "A portable, autonomous, urban reconnaissance robot," in *Sixth International Conference on Intelligent Autonomous Systems*, (Venice, Italy), July 2000.
2. N. Kehtarnavaz, N. C. Griswold, and J. S. Lee, "Visual control of an autonomous vehicle (bart) - the vehicle following problem," *IEEE Transactions on Vehiclular Technology* **40**, pp. 654–662, August 1991.
3. B. Motazed, "Measure of the accuracy of navigational sensors for autonomous path tracking," in *Proceedings of SPIE, 2058*, pp. 240–249, 1994.
4. R. O. Allen and D. H. Chang, "Performance testing of the systron donner quartz gyro." JPL Engineering Memorandum, EM #343-1297, January 5, 1993.
5. R. L. Farrenkopf, "Analytic steady-state accuracy solutions for two common spacecraft estimators," *Journal of Guidance and Control* **1**, pp. 282–284, July-Aug. 1978.
6. E. J. Lefferts and F. L. Markley, "Dynamics modeling for attitude determination," *AIAA Paper 76-1910* , Aug. 1976.
7. J. R. Wertz, ed., *Spacecraft Attitude Determination and Control*, vol. 73 of *Astrophysics and Space Science Library*, D. Reidel Publishing Company, Dordrecht, The Netherlands, 1978.
8. P. S. Maybeck, *Stochastic Models, Estimation and Control*, vol. 141-1 of *Mathematics in Science and Engineering*. Academic Press, 1979.
9. G. M. Siouris, *Aerospace Avionics Systems*, ch. 5. Academic Press, Inc., 1993.
10. Y. Fuke and E. Krotkov, "Dead reckoning for a lunar rover on uneven terrain," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pp. 411–416, 1996.
11. P. Bonnifait and G. Garcia, "A multisensor localization algorithm for mobile robots and it's real-time experimental validation," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pp. 1395–1400, 1996.
12. J. Stuelpnagel, "On the parameterization of the three-dimensional rotation group," *SIAM Rev.* **6**, pp. 422–430, Oct. 1964.
13. J. C. K. Chou, "Quaternion kinematic and dynamic differential equations," *IEEE Transactions on Robotics and Automation* **8**, pp. 53–64, Feb. 1992.
14. J. J. Craig, *Introduction to Robotics*, ch. 2, pp. 55–56. Addison-Wesley, 2nd ed., 1989.
15. E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics* **5**, pp. 417–429, Sept.-Oct. 1982.
16. S. I. Roumeliotis, *Robust Mobile Robot Localization: From single-robot uncertainties to multi-robot interdependencies*. PhD thesis, Electrical Engineering Department, University of Southern California, Los Angeles, California, May 2000.
17. S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, "Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1656–1663, (Detroit, MI), May 10-15 1999.
18. K. N. Murphy, "Analysis of robotic vehicle steering and controller delay," in *Fifth International Symposium on Robotics and Manufacturing (ISRAM)*, pp. 631–636, (Maui, Hawaii), August 1994.
19. A. Kelly, "A feedforward control approach to the local navigation problem for autonomous vehicles," Tech. Rep. CMU-RI-TR-94-17, Carnegie Mellon University, 1994.
20. O. Amidi and C. Thorpe, "Integrated mobile robot control," in *Proceedings of SPIE, 1388*, pp. 505–523, 1990.
21. A. Ollero and G. Heredia, "Stability analysis of mobile robot path tracking," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 461–466, 1995.
22. A. L. Rankin, C. D. Crane, and D. Armstrong, "Evaluating a PID, pure pursuit, and weighted steering controller for an autonomous land vehicle," in *Proceedings of SPIE, 3210*, pp. 1–12, 1997.
23. K. Astrom and T. Hagglund, PID Controllers: Theory, Design, and Tuning, International Society for Measurement and Control,, Research Triangle Park, N.C., 2nd ed., 1995.